

# **Web Service Integration Report**

## **1. Introduction**

The objective of this assignment is to integrate external web services into a backend application using secure and structured methods. The project focuses on consuming the Open Weather Map API to fetch current weather data for both single and multiple cities. The assignment aims to develop understanding of web APIs, service integration, authentication mechanisms, and secure API communication.

## **2. Implementation Steps**

The major development and configuration steps completed in this assignment include:

1. Cloning the collaborative GitHub repository for team development.
2. Installing project dependencies using 'npm install'.
3. Creating and configuring an .env file containing sensitive API keys and server configuration values.
4. Implementing Express routes, controllers, and service layers for fetching weather data.
5. Integrating JWT authentication middleware to protect API endpoints.
6. Testing the application thoroughly using Postman to verify successful responses.
7. Handling errors, invalid inputs, and network failures using custom middleware and try/catch logic.

## **3. Tools and Technologies Used**

- Node.js (server-side JavaScript runtime)
- Express.js (backend web framework)
- Axios (HTTP client for API consumption)
- JSON Web Tokens (JWT) for authentication and security
- Postman (API testing platform)
- GitHub (version control and collaboration)
- Visual Studio Code (development environment)

## **4. Results and Observations**

- JWT token generation works correctly.
- Weather data retrieval for single and multiple cities was successful.
- Invalid cities produce error responses and are logged in logs/errors.log
- Postman tests confirmed authentication, headers, and API functionality.

## 4.1 Screenshot: Successful Single-City API Response

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections, environments, and flows. The main workspace displays a 'Weather API Integration - Group 9 / Weather Endpoints / Get Weather - Single City' collection. A GET request is selected with the URL {{baseUrl}} /api/weather/:city. The 'Body' tab shows a JSON response for 'city': 'London'. The response body is as follows:

```
1 {  
2   "success": true,  
3   "data": {  
4     "city": "London",  
5     "country": "GB",  
6     "coordinates": {  
7       "lat": 51.5008,  
8       "lon": -0.1257  
9     },  
10    "temperature": {  
11      "current": "3.73",  
12      "feels_like": "0.55",  
13      "min": "2.79",  
14      "max": "4.86",  
15      "unit": "celsius"  
16    }  
17  }  
18 }
```

The status bar at the bottom indicates a 200 OK response with 741 ms and 642 B.

## 4.2 Screenshot: Successful Multiple-City API Response

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with various collections, environments, flows, and files. The main workspace shows a POST request for 'Get Weather - Multiple Cities' with the URL `[[baseUri]] /api/weather/multiple`. The request body contains a JSON object with an array of city names: London, Paris, Tokyo, New York, and Sydney. The response tab shows a 200 OK status with a response time of 1.50 s and a size of 2.36 KB. The response body is a JSON object indicating success with a count of 5 results, each containing a city name and its country.

```
1 {
2   "cities": [
3     "London",
4     "Paris",
5     "Tokyo",
6     "New York",
7     "Sydney"
8   ]
9 }
```

```
1 {
2   "success": true,
3   "count": 5,
4   "results": [
5     {
6       "city": "London",
7       "success": true,
8       "data": {
9         "city": "London",
10        "country": "GB",
11      }
12    }
13  ]
14}
```

## 4.3 Screenshot: JWT Token Generation (Login)

The screenshot shows the Postman interface with the following details:

- Header Bar:** Home, Workspaces, API Network, Search Postman, Ctrl + K, Invite, Notifications, Upgrade.
- Left Sidebar (My Workspace):**
  - Collections: openapi, Oromia football federation, RESTful API Basics #blueprint, Telebirr, training, Weather API Integration - Group 9 (selected), Authentication (selected), GET Get Token (selected).
  - Environments.
  - History.
  - Flows.
  - APIs.
  - Files (BETA).
- Request Details:** Method: GET, URL: {{baseUri}} /api/auth/token.
- Params Tab:** Shows Query Params table with columns: Key, Value, Description, Bulk Edit.
- Body Tab:** Response status: 200 OK, 23 ms, 432 B. Body content:

```
{ "success": true, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJyb2xlIjoic3RlZGVudCIsImhlhdCI6MTc2NDQ1NzE0MiwiZXhwIjoxNzY0MDYwNzQyfQ. Ui6g-rds_CF_GLB1bgx2QW9xOn0Zj8hpRe9sFIes1Bw", "expires_in": "1h" }
```
- Bottom Bar:** Cloud View, Find and replace, Console, Import Complete, Postbot, Runner, Start Proxy, Cookies, Vault, Trash, 20°C Sunny, ENG, 10:52 AM.

## 4.4 Screenshot: Authorization Header (Bearer Token)

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar is visible, containing collections like 'openapi', 'Oromia football federation', 'RESTful API Basics #blueprint', 'Telebirr', 'training', and 'Weather API Integration - Group 9'. Under 'Weather API Integration - Group 9', there are three main sections: 'Authentication' (containing 'Get Token'), 'Weather Endpoints' (containing 'Get Weather - Single City', 'POST Get Weather - Multiple Cities', and 'GET Get Weather - Invalid City (Err...)'), and 'Error Cases' (containing 'GET Unauthorized - No Token', 'GET Unauthorized - Invalid Token', 'POST Bad Request - Empty Cities Ar...', and 'GET Health Check'). The 'Error Cases' section is currently selected.

The main workspace shows a single API request for 'Unauthorized - No Token'. The method is 'GET', the URL is '({baseUri}) /api/weather/London', and the response status is '400 Bad Request'. The response body is a JSON object:

```
1  {
2   "success": false,
3   "message": "Missing Authorization header",
4   "stack": "BadRequestError: Missing Authorization header\n    at validateAuthHeader (file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/src/utils/validation.js:33:11)\n    at verifyToken\n(file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/src/middleware/auth.js:25:19)\n    at Layer.handleRequest\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/lib/layer.js:162:17)\n    at next\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/lib/layer.js:157:13)\n    at Route.dispatch\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/lib/route.js:117:3)\n    at handle\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/index.js:438:11)\n    at Layer.handleRequest\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/lib/layer.js:162:17)\n    at C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/index.js:299:15)\n    at param\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/index.js:600:14)\n    at param\n(C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/node_modules/router/index.js:610:14)"}
```

The bottom of the screen shows the Windows taskbar with various pinned icons and system status information.

## 4.5 Screenshot: Empty Cities Array

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, environments, history, flows, and files. The main workspace shows a collection named "Weather API Integration - Group 9". A specific POST request titled "Bad Request - Empty Cities Array" is selected. The request URL is `POST {{baseUrl}} /api/weather/multiple`. The response status is 400 Bad Request, with a duration of 29 ms and a size of 1.66 KB. The response body is a JSON object:

```
1 {  
2   "success": false,  
3   "message": "Please provide a 'cities' array in the request body.",  
4   "stack": "BadRequestError: Please provide a 'cities' array in the request body.\n      at validateCitiesArray  
      (file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/src/utils/validation.js:10:11)  
      |  
      | at file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/src/routes/weatherRoutes.  
      |js:41:8|  
      | at file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/src/utils/errors.  
      |js:62:19|  
      | at Layer.handleRequest  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\lib\\\\layer.  
      js:162:17)|  
      | at next  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\lib\\\\route.  
      js:167:13)|  
      | at verifyToken (file:///C:/Users/dawit/Desktop/Coding/web-service-integration-assignment/  
      |src/middleware/auth.js:29:5)|  
      | at Layer.handleRequest  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\lib\\\\layer.  
      js:162:17)|  
      | at next  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\lib\\\\route.  
      js:162:17)|  
      | at Route.dispatch  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\lib\\\\route.  
      js:117:3)|  
      | at handle  
      (C:\\\\Users\\\\dawit\\\\Desktop\\\\Coding\\\\web-service-integration-assignment\\\\node_modules\\\\router\\\\index.  
      js:436:11)"  
5 }
```

The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray indicating it's 20°C, sunny, and the time is 10:55 AM.

## 4.6 Screenshot: Error Log

The screenshot shows the VS Code interface with the 'logs' folder selected in the Explorer sidebar. The 'errors.log' file is open in the center editor area, displaying a log of errors from November 25, 2025. The errors are related to failed requests to an external API for weather data. The terminal at the bottom shows a single command: 'curl -s https://api.example.com/weather/multiple | jq .'. The status bar indicates the command was run at 10:55:08.

```
1 2025-11-25T06:46:35.035Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
2 2025-11-25T06:46:35.043Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
3 2025-11-25T06:46:42.319Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
4 2025-11-25T06:46:42.321Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
5 2025-11-25T06:48:53.896Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
6 2025-11-25T06:48:53.907Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
7 2025-11-25T06:49:05.316Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
8 2025-11-25T06:49:05.318Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
9 2025-11-25T06:50:11.630Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
10 2025-11-25T06:50:11.639Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
11 2025-11-25T06:50:24.303Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
12 2025-11-25T06:50:24.312Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
13 2025-11-25T06:54:55.407Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
14 2025-11-25T06:54:55.409Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
15 2025-11-25T06:56:00.503Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
16 2025-11-25T06:56:00.513Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
17 2025-11-25T06:56:29.200Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
18 2025-11-25T06:56:29.202Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
19 2025-11-25T06:58:22.356Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
20 2025-11-25T06:58:22.358Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
21 2025-11-25T06:58:27.012Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
22 2025-11-25T06:58:27.013Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
23 2025-11-25T06:59:08.401Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
24 2025-11-25T06:59:08.415Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
25 2025-11-25T06:59:17.845Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
26 2025-11-25T06:59:17.848Z [ERROR] GET /weather/multiple -> Failed to fetch weather from external API
27 2025-11-25T07:09:28.575Z [ERROR] fetchWeatherByCity(multiple) failed: Request failed with status code 404
```

## 5. Conclusion

This project successfully fulfills all requirements of the Web Services Integration assignment. It demonstrates secure API integration, proper error handling, data transformation, and logging.

This assignment offered hands-on experience in integrating external web services into a backend system using Node.js and Express. Through the implementation of JWT authentication, secure API communication, and structured project architecture, we gained practical understanding of service integration and backend development. The collaborative workflow on GitHub also provided real-world exposure to version control, branch management, and team-based software development practices.